Jonathan W. Valvano March 1, 2013, 10:00 to 10:50am

(10) Question 0. Your crib sheet will be graded on content and correctness.

(10) Question 1. Time to wait as it ages is $m*t_1$. Time to wait as it waits to be scheduled $(n-1)*t_2$. Total time is sum of the two waits $m*t_1+ (n-1)*t_2$

(5) Question 2. The four most important factors for effective debugging are control (stabilization), observability (dumps), nonintrusiveness (fast), and coverage (testing the cases that matter).

(20) Question 3. Select the best term that describes each definition.

A formal model that can be used to study data flow where nodes are connected using FIFO queues, such that we can guarantee that the FIFOs never become full. **Kahn Process Network**

Software execution that cannot be divided or interrupted. Once started, the operation will run to its completion without interruption. **atomic**

The condition where once a thread begins to wait on a resource, there are a finite number of threads that will be allowed to proceed before this thread is allowed to proceed. **bounded waiting**

A process where a governing body (e.g., FDA, FCC, DOD etc.) gives approval for the use of the device. It usually involves demonstrating the device meets or exceeds safety and performance criteria. **certification**

A scheduling algorithm with round robin order but varying time slice. If a thread blocks on I/O, its time slice is reduced. If it runs to completion of a time slice, its time slice is increased. **exponential queue**

A scenario that occurs when two or more threads are all blocked each waiting for the other with no hope of recovery. **deadlock**

An indirect function call added to a software system that allows the user to attach their programs to run at strategic times, created at run time and do not require recompiling the entire system. **Hook**

A characteristic when the presence of the collection of information itself does not affect the parameters being measured. **nonintrusive**

A software technique to guarantee subfunctions within a module are executed in a proper sequence. For example, it forces the user to initialize I/O device before attempting to perform I/O. **path expression**

The percentage of resource utilization below which the RTOS can guarantee that all deadlines will be met. **Breakdown utilization**

(10) Question 4. *There was a typo in the exam, so minute and seconds where reversed.*

Part a) The critical section occurs in the multistep, nonatomic read of `Min`, then `Sec`.

58:02
59:02
00:02 reads `Min` getting the 2, interrupt occurs, reads `Sec` getting the 00
00:03
01:03

Part b) The critical section is defined as a sequence of code during which if an interrupt or context switch were to happen then something bad happens, like crash, lost data, extra data, bad data. The critical section must in the foreground, because the background interrupts the foreground, not the reverse. The critical section must begin and end with memory accesses to shared globals.

```
0x00000128          Foreground
0x00000128 B510          PUSH   {r4,lr}
0x0000012A 48FE          LDR    r0,[pc,#1016]  ;address of Min
0x0000012C 6802          LDR    r2,[r0,#0x00]  ;value of Min    ← beginning
0x0000012E 48FE          LDR    r0,[pc,#1016]  ;address of Sec
0x00000130 6801          LDR    r1,[r0,#0x00]  ;value of Sec    ← end
0x00000132 A0FE          ADR    r0,{pc}+2      ;pointer to string
0x00000134 F000FFD2      BL     printf
0x00000138 BD10          POP    {r4,pc}
0x0000013A          Timer0A_Handler
0x0000013A 2004          MOVS   r0,#0x04       ;TIMER_ICR_CAECINT
```

(20) Question 5. Create a variable describing the state of the system and a mutex semaphore. A couple of ways I look at semaphore applications to quickly tell if the solution makes sense

- One cannot execute an if-then statement based on a global unless there is mutual exclusion
- One cannot have two accesses to globals unless there is mutual exclusion
- Each wait must be paired with a signal

```
int bNeedInitialization=1;  // variable, meaning not initialized
int Mutex=1;         // binary semaphore with initial value
void SysTick_Init(void){
  OS_Wait(&Mutex);
  if(bNeedInitialization){
    bNeedInitialization = 0;  // prevents from initializing twice
    NVIC_ST_CTRL_R = 0;             // disable SysTick
    NVIC_ST_RELOAD_R = 0x00FFFFFF;  // maximum reload value
    NVIC_ST_CURRENT_R = 0;          // clears SysTick counter
    NVIC_ST_CTRL_R = 0x05;      // enable SysTick with core clock
  }
  OS_Signal(&Mutex);
}
```

(25) Question 6. Write the assembly code for the PendSV handler.

```
PendSV_Handler                ; Saves R0-R3,R12,LR,PC,PSR
    CPSID   I                 ; Prevent interrupt during switch
    PUSH    {R4-R11}          ; Save remaining regs r4-11
    LDR     R0, =RunPt        ; R0=pointer to RunPt, old thread
    LDR     R1, [R0]          ;    R1 = RunPt
    STR     SP, [R1,#4]       ; Save SP into TCB
    LDR     R2, =ActivePt     ; R0=pointer to ActivePt,
    LDR     R1, [R2]          ; R1 = ActivePt
    STR     R1, [R0]          ;    RunPt = R1
    LDR     SP, [R1,#4]       ; new thread SP; SP = RunPt->sp;
    POP     {R4-R11}          ; restore regs r4-11
    CPSIE   I                 ; tasks run with interrupts enabled
    BX      LR                ; restore R0-R3,R12,LR,PC,PSR
```